APL*PLUS®/PC

TOOLS

Volume 1

Technical Summary

This document was prepared to assist users of STSC's APL*PLUS Service and software products; its contents may not be used for any other purpose without written permission. The material contained herein is supplied without representation or warranty of any kind. STSC, Inc., therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this document or the material contained herein.

Copyright © 1984 STSC, Inc.

Printed in the United States of America.

All rights reserved, including the right to reproduce this document or any portion thereof in any form.

APL*PLUS is a service mark and trademark of STSC, Inc., registered in the United States Patent and Trademark Office and in certain other countries.

IRMA and Decision Support Interface are trademarks of Digital Communications Associates, Inc.
IBM is a registered trademark of International Business Machines Corporation.

APL*PLUS/PC TOOLS

Volume 1

Features and Capabilities

APL*PLUS/PC TOOLS, Volume 1, is a comprehensive collection of application-building subsystems and productivity aids for APL programmers using the APL*PLUS/PC Application Development System. The product includes:

IRMA™ Board Support

A customized version of the IRMA terminal emulator allowing a PC to act as a 3278 or 3279 terminal connected to an IBM® mainframe computer.

- Keyboard layout of the APL*PLUS/PC System.
- Support for the APL character set.
- APL printer support for the pseudo-327x terminal using the EPSON FX-series printers.

High-speed file transfer between the APL*PLUS/PC System and a mainframe VS APL system running under VM/CMS or a mainframe APL*PLUS System (1000 series). Top-level functions can be adapted to an MVS/TSO environment.

- PC DOS or APL*PLUS component files on the PC, CMS or APL*PLUS component files on the mainframe. Special handling of APL object transfer files.
- Second-level transfer subroutines for custom data transfer applications for VM/CMS and MVS/TSO mainframes.
- Data conversion, blocking, transmission, and verification.

Functions to interface with the IRMA board under APL program control.

- Access to the 3278 screen and PC control of mainframe applications.
- Prototype function to move mainframe information to the PC through the medium of the screen.

PC as a Host Computer

An environment where one PC running the APL*PLUS/PC System is configured to act as a host, mainframe-like computer. Terminals or other PCs with the APL*PLUS/PC System are connected to the host PC to run applications using standard APL input and output methods requiring no "operator" intervention.

A data transfer facility for this environment where data variables or entire files can be moved between PCs when one of them is configured as a host.

- Transfer of PC DOS and APL*PLUS component files. Special handling of APL object transfer files.
- Data conversion, blocking, transmission, and verification.

Screen Management

Functions to build, edit, and maintain application menus, and then present them to, handle interactions with, and interpret responses of the end user.

- Extremely varied menu styles, from a single line of choices to a multi-page, multi-choice offering.
- Support for monochrome, color, and other attributes.
- Support for "bounce bar" selection, "write in" choices, or both.
- Utilities to edit "source" menus and dynamically alter "object" menus.

Functions to build, edit, and maintain full-screen input forms, and then present them to, handle interations with, and interpret and verify responses of an application user.

- Support for monochrome, color, and other attributes.
- Utilities to query, test, and dynamically alter field information.
- Management of multiple input forms.
- Standard and user-defined field types.

Reporting and Output Formatting

Functions to direct output to screen, printer, APL file, or DOS file.

- Page numbers tracked and positioned at top or bottom.
- Placement of titles and footers on each page.
- Prevention of the splitting of related output at page bottoms.
- Facility for character translations.
- Deferred printing of reports directed to file.

Disk Emulation

Software device driver to emulate a physical disk in RAM.

RAM disk management from the APL environment.

- Functions to set and query the size and memory address.
- Functions showing typical uses.

Software Development Aids

Utilities to analyze, modify, and maintain APL functions.

- Localization of subfunctions, reordering of local identifiers.
- Removal of embedded assignments, removal of superfluous parentheses.
- Sequential renaming of labels, or their removal.
- String search and replacement.
- Diamondization or undiamondization of multiple statements.
- Insertion, alignment, or removal of comments.
- Identifier classification and cross-reference displays.

Utilities to analyze, modify, and maintain workspaces.

- Environment, storage, and system variable synopses.
- Function and variable synopses.
- Full or partial search for strings, either as syntactic or arbitrary function elements.
- Stand-alone function lister.
- Function dependency analysis.
- Locked and unlocked function lists.
- Expanded information about the state indicator and identifier classifications.

File image of a workspace.

- Functions to search, analyze, modify, and maintain functions on file.
- Re-creation of a workspace from its file image.

Workspace documentation facility.

- Workspace, function, and variable synopses.
- Function listings and identifier cross-references in various formats.
- Function classifications and interdependencies.
- Workspace cross-reference and dependency tree displays.
- Output to screen, printer, or file.

Workspace comparison facility.

- Line-by-line differences in the functions in two workspaces.
- Lists of absent or identical objects in the other workspace.
- Output to screen, printer, or file.
- Stand-alone comparison of individual functions in workspace or file.

Games and Recreation

Reversi -- Strategy, capture, and mercurial shifts of fortune.

Three-dimensional Tic-Tac-Toe -- Solitaire play against the computer.

Product Contents

A megabyte of program software on three disks.

- Approximately 700 APL functions.
- Auxiliary APL and DOS files.
- "Bootstrap" process for the mainframe end of the IRMA-based data transfer facility.

300 page reference manual in a linen binder with slipcase.

System Requirements

APL*PLUS/PC System, Release 3.0 or later.

IBM PC DOS, Version 2.00 or later.

Minimum of 256 kilobytes of RAM.

For those wanting to use the IRMA support, an installed IRMA Board and accompanying software, Version 1.25 or later.

Example

Here is an illustration (for a PC with disk drives A and B) of the installation of the small RAM disk at a size just big enough to contain the DOS command processor and the IRMA terminal emulator, and a 200 kilobyte RAM disk to hold scratch files.

- Bring up APL and load the *RAMDISK* workspace from your working copy of APL*PLUS/PC TOOLS Disk 1.

- Decide what names you want for your RAM disks, and make sure your *CONFIG.SYS* file references them. This example uses the name *RAMDSK*50 for a 50 kilobyte RAM disk as disk C, and *RAMDSKLG* as disk D. Execute:

  ```
  DOSEDIT 'CONFIG.SYS'
  ```

  and insert the lines

  ```
  DEVICE=RAMDSK50.COM
  DEVICE=RAMDSKLG.COM
  ```

  into the *CONFIG.SYS* file.

- Initialize their names and sizes (and locations).

  ```
  'RAMDSK50' RDINIT 50
  'RAMDSKLG' RDINIT 200
  ```

- Use *DOSEDIT* again to create or edit your *AUTOEXEC.BAT* file to contain the following statements:

  ```
  COPY TERM3278.EXE C:
  COPY COMMAND.COM C:
  COMMAND C:
  ```

  The first two statements copy the IRMA terminal emulator and the DOS command processor from your system boot disk to the RAM disk when *AUTOEXEC.BAT* is executed. The third statement invokes the command processor that is now found on disk C (the small RAM disk). If you extended the third statement to read

  ```
  COMMAND C: /CC:TERM3278
  ```

  the IRMA terminal emulator would be invoked during the setting of the command file.

- Re-boot your revised system disk to activate the RAM disks.

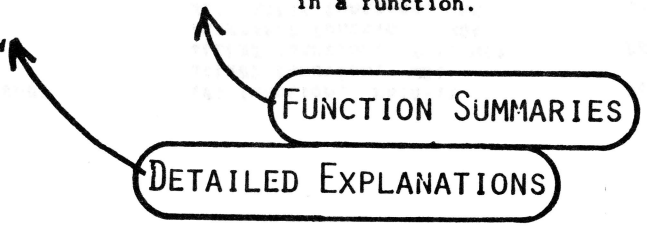
- Thereafter, every time you boot your system, the RAM disks are activated, the system looks for *COMMAND.COM* on disk C whenever necessary (for example, when invoking DOS from APL), and the IRMA terminal emulator can be loaded from disk C (with the DOS command *C:TERM*3278). When creating and referencing temporary APL*PLUS/PC component files, use library number 3 (drive D, the large RAM disk). Libraries 0 and 1 still refer to files on the actual A and B drives.

- To copy (and rename) a temporary APL file from the RAM disk D to the permanent disk A, you could enter the command

  ```
  COPY  D:TEMPFILE.ASF  A:PERMFILE.ASF
  ```

Function Summary

ALLFNS	Returns all functions in the workspace. Detailed documentation is in *"WSCARE"*.
BRKOUT	Modifies a function to break out embedded assignments.
BY	Companion function to *FNREPL* that indicates the string to replace and its replacement value. Documented with *FNREPL*.
DIAM	Inserts diamond statement separators into a function, where possible.
FCL	Acts as a general purpose function cleanser.
FNIDS	Searches a function for identifiers that are locals, labels, direct assignments, indexed assignments, or quad names.
FNREPL	Replaces one string by another in functions.
LAMP	Inserts or removes comment symbols from a function.
LAMPALIGN	Aligns comments within a function.
LOCALIZE	Localizes specified subfunctions within a function.
ORDLOC	Reorders the local identifiers in the header of a function.
RELABEL	Modifies a function to use the set of labels *L1*, *L2*, ...
SELDIAM	Selectively inserts diamonds in a function.
SNUFF	Removes comments from a function.
UNDIAM	Removes diamonds from a function by placing multiple statements on multiple lines.
UNLABEL	Removes labels from a function.
UNPAREN	Removes superfluous parentheses from a function.
XREF	Displays a cross-reference of the identifiers in a function.



Workspace: *WSCARE* **Function:** *USEDBY*

The function *USEDBY* returns the list of the global objects in the workspace that are referenced by a list of unlocked functions.

Syntax

namelist ← *USEDBY* fnlist

The argument is a list of function names. The list can be a character matrix with one name per row, or a character vector with names separated by spaces.

The result is a matrix of function and variable names whose definitions are global to any of the functions in the right argument, including the functions in fnlist. The names are in alphabetical order, one name per row.

If none of the functions in the right argument exist, the result is a matrix with no rows.

Effect

Objects referenced in comments and character constants are not included in the result. Likewise, objects that are global to a function but not currently in the workspace are not included.

The list of names in the result includes objects that are referenced indirectly. For example, if a function *PUTDATA* calls *CHECKDATA*, and *CHECKDATA* calls *CHECKFIELDS*, then the result of *USEDBY 'PUTDATA'* includes *CHECKDATA*, *CHECKFIELDS*, and *PUTDATA*.

Examples

```
      USEDBY 'ADDRECORD DELETERECORD'
ADDRECORD
CHECKADD
CHECKDELETE
DELETERECORD
FILE
GETRECORD
PUTRECORD

      )ERASE ALLBUT USEDBY 'DRIVER'
```


DETAILED FUNCTION DESCRIPTIONS

COMPREHENSIVE INDEX

AF8 function, PX10-11
ALLBUT function, WS3
ALLFNS function, FN1, WS3
ALLFNSBUT function, WS4
ALLVARSBUT function, WS4
APL files, see Files
APL mode, IT6, IT10
APL*PLUS/PC System, version needed, Introduction 1
AP110, IX15, IX22, IX25
Attributes
  menu, MC3, MC19
  on 3270, IF7, IF14, IF16, IF19
  screen, SC4-5, SC10, SN27

Bootstrapping to the mainframe, IX2
BRKOUT function, FN3
BUILDMENU function, MC18
BY function, FN12-13

CHOICEFROM function, IF2, SC8, SN8, SN11, SN21
CMS, IX15, IX22, IX25
Comments in a function
  aligning, FN16, ST7, WD5
  constructing, FN8, FN14
  removing, FN4, FN8, FN21
  special, FN8, WC2
Communications, see *I78FNS*, *I78TERM*, *I78XFER*, *PCHOST*, *PCPCXFER*
Contents of the disks, Introduction 3
Copying portions of APL*PLUS/PC TOOLS, Introduction 4-5
Cross-referencing
  functions, FN25, ST4, WD7
  workspaces, ST4-12, WD5-12, WS20
Cursor, IF9, MC7-8, SN29
Customizing the IRMA terminal emulator, IT3-4

Data transfer, also see *I78XFER*, *PCPCXFER*
  as character vector, IX5-7, PX5-7
  from file to file, IX3-4, PX2-4
  functions for, IX7-8, PX7-8

DEREP function, IX7, IX10, PX7
DESCRIBE function, Introduction 2, 5
DIAM function, FN4
Diamonds in a function
  adding, FN4, FN7, FN20
  removing, FN7-8, FN22
Disk emulator, also see *RAMDISK*
  configuring, RD2-4, RD7
  creating, RD3, RD7
  defined, RD1
  functions for using, RD5
  initializing, RD7
  installation, RD2-4
  moving files from, RD4
  uses, RD1
Documentation, see also Workspace documentation Introduction 1-3
DOS
  AUTOEXEC.BAT, RD2-4
  COMMAND.COM, RD1, RD4
  CONFIG.SYS, RD2-4
  COPY.COM, IT2-4, RD3-4
  device drivers, RD1-2
  file editing, RD6
  FORMAT.COM, IT3
  MODE.COM, PH4
  version supported, Introduction 1
Downloading APL characters, IT5
DTF function, IX11
DTFALL function, IX11

Emulator, see Disk emulator, Terminal emulator
EMPTYMENU function, MC19-20
ENVIRONMENT function, WS5
Error messages, IF10, IX7, IX26, IX29, PX7-9
Examples
  data transfer, IX5-7, PX5-7
  file transfer, IX4, PX3-4, WD17
  function comparison, WC3
  GRABDATA function, IF1-3
  installing RAM disks, RD3-4
  ORDER function in the host facility, PH5-6
  ORDERS function in the screen facility, SN6-7

FN=*FNCARE* FT=*FILETOWS* IF=*I78FNS* IT=*I78TERM* IX=*I78XFER*
MC=*MENUCARE* MN=*MENU* OP=*OUTPUT* PH=*PCHOST* PX=*PCPCXFER*
RD=*RAMDISK* RV=*REVERSI* SC=*SCRNCARE* SN=*SCREEN* ST=*WDSTATE*
TT=*TTT3D* WC=*WSCOMP* WD=*WSDOC* WF=*WFCARE* WS=*WSCARE*
WT=*WSTOFILE*

Overview of the Menu Facility

A menu, whether in a restaurant or application, presents a list of choices. A computer program uses a menu to elicit a response from the user. The program then evaluates and acts on the choice.

By using menus, your programs can use a full-screen approach to prompting for input. Such an approach lets you include far more prompting information. Instead of having a half-line prompt asking for a "Y" or an "N", you can spell out all the possible choices. You can also show the user how the application is progressing, provide background information, and so on. You can even reduce a whole series of "yes-no" questions to one choice from an appropriate list.

The Extended Menu

The traditional full-screen menu displays a list of choices, with one highlighted in reverse video. The user chooses from the list by moving the highlighting with the cursor keys and pressing Enter.

Your menu can overlay all or part of the current screen contents and then restore the previous screen. You need not limit your menu to the amount of unused space on the screen, or use it only at a break point between different screenfuls of output. Your menu can act as a temporary extra screen for fuller communication with your user.

Instead of restoring the screen when you are through, you can choose to leave the menu in view or even alter the window contents and cursor location when done with the menu.

The functions in the menu facility enable you to:

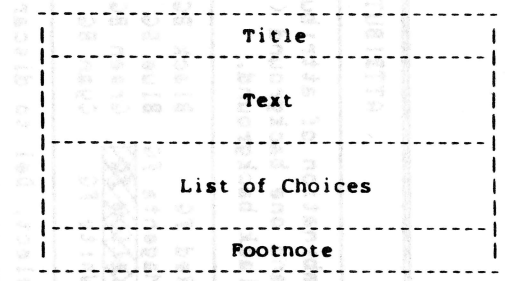
- present information along with the choices
- present information without any choices, and wait for the user to press any key
- let the user pick more than one of the choices
- return either the numbers of the choice or choices or return a character string associated with each of the choices
- let the user "write-in" a choice not found in the list on the menu
- allow your menu to "time out" if the user enters no selection
- substitute programmed function keys for cursor keys to move the highlighting when the application requires a numeric pad.

The functions also allow you to decorate your menu by using screen attributes like color, intensity, or flashing to stress a word or sentence; or by putting boxes around the menu on the screen and subdivisions within the box.

The menu can have as many as four sections:

- a title area, to identify the nature of the information to be presented or the choice to be made (each line can be flush left, flush right, or centered)
- a text area, for information on the choices or information only (text is automatically wrapped at word boundaries)
- a choice list (automatically formatted)
- a footer, typically used for giving short directions and/or accepting write-in answers (each line can be flush left, flush right, or centered).

A menu with all four parts would look similar to the following diagram.

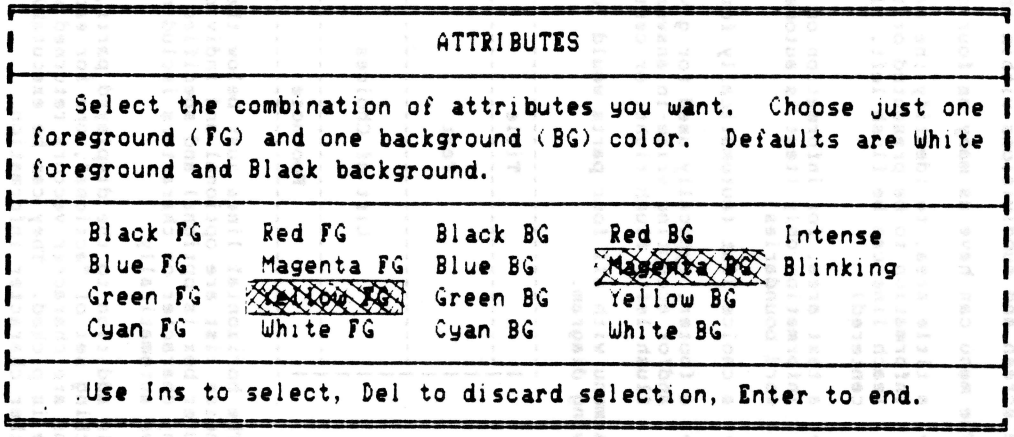


The horizontal lines just below the title, the text, and the choice list are optional and individually specifiable. The outer box is optional and specified as a whole. You can use any character or characters including spaces. Corners are handled automatically.

In addition to the displayed parts, your menu can include a matching set of "actions", one for each choice. These actions are character vectors returned when the corresponding choice is picked. They can be executable APL expressions or any other character information.

How to Use the Menu Facility

1) Design the menu as you want it to appear in your application



2) CONSTRUCT THE SOURCE MENU BY USING THE INTERACTIVE BUILDMENU FUNCTION OR BY USING A COMBINATION OF THE EMPTYMENU FUNCTION AND THE FULL SCREEN EDITOR

```
     ATTRsrc
|:g@
| 1. Menu Comment----------:MENU FOR ATTRIBUTE SETTING
| 2. Window Specification--:0 5 25 70
| 3. Full or Minimum Height:0
| 4. Surrounding Box Type--:-2
| 5. Title Text------------:ATTRIBUTES
| 6. Title Separator Line--:-1
| 7. Text Explanation------:   Select the combination of attributes you
      want.  Choose just one foreground (FG) and one background (BG) c
     olor.  Defaults are White foreground and Black background.
| 8. Text Separator Line---:-1
| 9. Choices/Actions Spec.-:1
|10. Number Choices Allowed:4
|11. Choices Separator Line:-1
|12. Footer Text-----------:Use Ins to select, Del to discard selection
     , Enter to end.
|13. Write-in User Entry---:0
|14. Save & Restore Window-:4
|15. Write-in Cursor Shape-:
|16. Menu Display Timeout--:
|17. PgUp & PgDn Returns---:
|18. Break & Escape Returns:
|19. Screen Attributes-----:
|20. Reserved Empty Segment:
|21. Choice 1--------------:Black FG
|22. Choice 2--------------:Blue FG
|23. Choice 3--------------:Green FG
|24. Choice 4--------------:Cyan FG
|25. Choice 5--------------:Red FG
|26. Choice 6--------------:Magenta FG
|27. Choice 7--------------:Yellow FG
|28. Choice 8--------------:White FG
|29. Choice 9--------------:Black BG
|30. Choice 10-------------:Blue BG
|31. Choice 11-------------:Green BG
|32. Choice 12-------------:Cyan BG
|33. Choice 13-------------:Red BG
|34. Choice 14-------------:Magenta BG
|35. Choice 15-------------:Yellow BG
|36. Choice 16-------------:White BG
|37. Choice 17-------------:Intense
|38. Choice 18-------------:Blinking
```

3) COMPILE THE SOURCE MENU INTO AN OBJECT MENU

```
ATTRobj ← MENUCOMPILE ATTRsrc
```

4) INCORPORATE THE MENU INTO YOUR APPLICATION PROGRAM

```
    ∇ R←ATTRIBUTECHOICE;A;B
[1]  ⍝ att←ATTRIBUTECHOICE -- Uses object menu ATTRobj
[2]  ⍝ to prompt for screen attribute selection.
[3]  ⍝ Corresponding source menu is variable ATTRsrc;
[4]  ⍝ that is, ATTRobj←MENUCOMPILE ATTRsrc.
[5]  ⍝ Make result zero for any MENU escape:
[6]  L1:R←0 ◊ A←MENU ATTRobj ◊ →(∨/A≥0)⍴0
[7]  ⍝ Test foreground selection:
[8]   B← 1 2 3 4 5 6 7 8 ∈A ◊ →(1≥+/B)⍴L2
[9]   ⎕←'*** Only 1 foreground color allowed' ◊ →L1
[10] ⍝ Default foreground attribute is 7:
[11] L2:R←1↑(B/ 0 1 2 3 4 5 6 7),7
[12] ⍝ Test background selection:
[13]  B← 9 10 11 12 13 14 15 16 ∈A ◊ →(1≥+/B)⍴L3
[14]  ⎕←'*** Only 1 background color allowed' ◊ →L1
[15] ⍝ Default background attribute is 0:
[16] L3:R←R++/ 16 8 128 ×(1↑B/ 0 1 2 3 4 5 6 7), 17 18 ∈A
    ∇
```

Introduction

The screen facility consists of tools designed to allow you to easily incorporate full-screen data entry and display into an application. The tools are organized into two workspaces: *SCRNCARE* and *SCREEN*. *SCRNCARE* contains an editor to assist you in designing and defining screen layouts. There is also a *SCRNHELP* file for use with the *SCRNCARE* workspace.

The *SCREEN* workspace is the operational part of the APL*PLUS/PC TOOLS screen facility. It contains utility functions that you integrate into your full-screen application. They help you:

- open defined screens; that is, enable them for use
- gather users' input from open screens
- dynamically change portions of screens
- evaluate, store, or discard the captured input
- manage a hierarchy of screens without losing partially complete user input.

The application workspace that uses the screen facility must contain the necessary functions from the *SCREEN* workspace and the variables you create containing the applicable screen definitions. You access the screen facility when you open (enable) one or more screens and it remains available until the last screen is closed. Additional workspace variables are defined to hold the images of open screens, including the contents of data fields and the position of the cursor for each screen.

Overview

When this reference manual refers to an entity called a screen, it means a single panel defined for use in the application. The screen consists of text, decoration (for example, borders and colors), and a set of fields. The text for a screen is never changed by either the application programs or the application user. The fields are areas on the screen whose contents can be altered by either the programs, the user, or both.

Fields are defined by their location on the screen. The fields have a name, a consistent display attribute for all positions in the field, and a field type. The field type indicates what verification is to be performed on entered data and how the system uses the field.

The system has four predefined field types:

Output -- The application programs can write in these fields, but the user cannot.

Character -- The user can enter any data.

Numeric -- The user can enter data that can be interpreted as one or more numbers.

Menu -- The user can enter data that matches an entry in a menu defined in the application, or the user can choose the entry from the menu itself on another system.

You can also define up to nine more field types of your own.

Using The Screen Facility

The steps for using the screen facility are as follows.

- Define the necessary screens using the *EDITSCREEN* function in the *SCRNCARE* Workspace. (See the "*SCRNCARE* workspace" for details on using this function.)
- Store the corresponding screen definition variables in a workspace or in a file.
- Move the appropriate screen definition variables into a copy of the *SCREEN* workspace.
- Open screens with the *SCOPEN* function, thus making them available to the other screen utilities.
- The *SCINPUT* function activates an open screen, displays it to the application user, and gathers and displays data that the user enters. After the input is gathered, standard or user-supplied validation routines can check the data.

The open screens can be modified dynamically based on other user input by the *SCPUTFLD*, *SCPUTFLDS*, and *SCPUTFLDATTR* functions. In the example, these functions would display appropriate information in the fields, depending on the type of work the user was doing. For example, if the user was changing existing data, all the information could be displayed, and the user would be allowed to type new information over the old.

Function Summary

CHOICEFROM	Presents a title and a list of choices and permits a single user selection. Detailed documentation is in "The *SCRNCARE* Workspace".
SCCLOSE	Releases a screen number and removes the screen definition from the open screen variables.
SCFKEYSOFF	Restores function key definitions, cursor position, and screen contents from saved global variables.
SCFKEYSON	Assigns current function key definitions, cursor position, and screen contents to global variables and then redefines the function keys for a screen input session.
SCFLDFIND	Returns a vector of field numbers corresponding to a list of field names for an open screen.
SCGETFLD	Returns the contents of a single field of an open screen.
SCGETFLDS	Returns the contents of multiple fields of an open screen.
SCHELP	Invokes the system help facility when the user presses Alt F6.
SCINIT	Initializes global variables used by the screen utilities.
SCINPUT	Displays and allows input to an open screen.
SCMENUCHOICE	Uses the choice list from a workspace global for a menu field to get desired choice from user pressing Alt F5.
SCMODIFIED	Determines if a specified field of the active screen has changed since starting input.
SCNAMES	Returns a matrix of the names of all open screens.
SCNUMS	Returns a vector of the screen numbers of all open screens.
SCOPEN	Associates a screen definition with a screen number, and makes that screen definition available for use by the screen utilities.
SCOUTPUT	Stores the contents of the active screen in the open storage variables, and activates (displays) an open screen.
SCPUTFLD	Stores data in a single field of an open screen.
SCPUTFLDATTR	Puts an attribute in a single field of an open screen.
SCPUTFLDS	Stores data in multiple fields of an open screen.
SCREAD	Brings into the active workspace from a file the screen definition global variables for a list of screens. Detailed documentation is in "The *SCRNCARE* Workspace".
SCREENFILER	Prompts and allows the user to store screen definitions on file, or to retrieve them from file. Refer to "The *SCRNCARE* Workspace".
SCSETCURSOR	Positions the cursor at the beginning of a field of an open screen.
SCSHOWMSG	Displays a message on the 25th line of the screen.
SCVALTYPE∆C	Standard validation function for character fields. (Documented with *SCVALTYPE∆1*.)
SCVALTYPE∆M	Standard validation function for menu fields. (Documented with *SCVALTYPE∆1*.)
SCVALTYPE∆N	Standard validation function for numeric fields. (Documented with *SCVALTYPE∆1*.)
SCVALTYPE∆1	Field validation function for user-defined field type 1, that is invoked as the user leaves each field of that type. The prototype distributed with APL*PLUS/PC TOOLS validates a certain style date field.
SCWRITE	Moves from the active workspace into a file the screen definition global variables for a list of screens. Detailed documentation is in "The *SCRNCARE* Workspace".
*SC*modified	Identifies all fields of the active screen that have changed since starting input to that screen.

Introduction

The *OUTPUT* workspace contains functions that direct output to the screen, to an APL file, to a DOS file, or to a printer; number pages, insert top and bottom titles and margins, group related output on a single page, and restart an interrupted report.

The appearance of reports is sometimes as important as the information in the reports. Formatting reports often takes more time and effort than calculations and data access routines. The system function ⎕*FMT* is a powerful tool for formatting numeric data, but it only does part of the job. You must still tack on titles and page numbers, handle page breaks, and so on. The functions in the *OUTPUT* workspace are as useful in printing output as ⎕*FMT* is in formatting it.

Report generation has three basic steps:

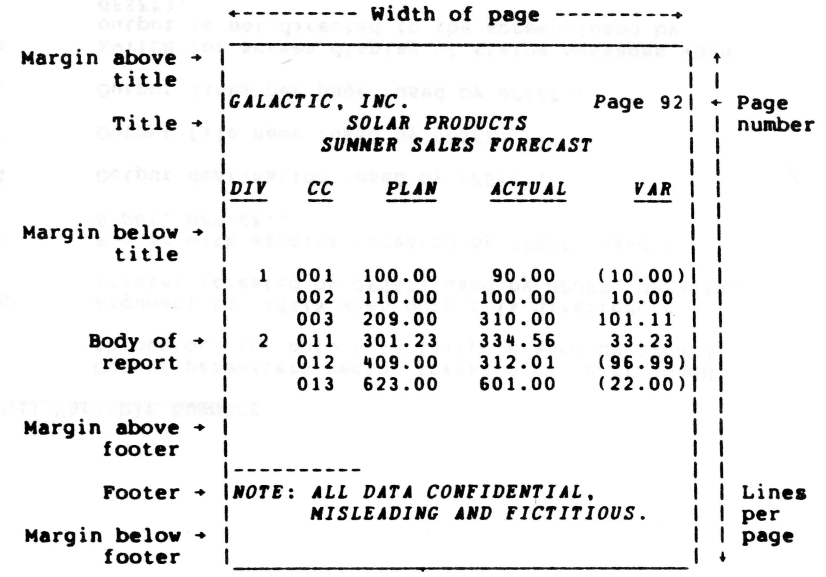
1. Setting up: determining page size and display area
2. Generating the output: formatting the data, skipping pages where necessary
3. Finishing up: erasing variables, resetting applications, and so on.

Procedure Summary

1. Set global variables as desired.
2. Call *OPSET* to initialize the process.
3. Call *OPTOP* and *OPBOT* to set the title and footer.
4. Generate the report with *OPOUT*, *OPBLOCK*, and/or *OPPAGE*.
5. Finish up by calling *OPEND*.

Example

```
    ∇ DEST REPORT DATA;FMT;PV;T;opb;opd;opl;opp;opt;⎕IO
[1]  ⍝ DATA is 5 column integer matrix:
[2]  ⍝   Division, Cost center, Plan ¢, Actual ¢, Variance ¢
[3]  ⍝ DEST is 1, 2, 3, or 4
[4]  ⍝ >>>>> Requires CENTER, COLNAMES, RWTD from FORMAT ws
[5]   ⎕IO←1 ⋄ opd←'SADP'[DEST] ⍝ establish report destination
[6]   opl←24 66 66 66 ⍝ lines per page for each destination
[7]   opp←5⍴0 ⍝ initialize page specs
[8]   opp[2]←1 0 0 0[DEST] ⍝ if screen, pause each page
[9]   opp[3]←3 ⍝ right justify page nos at top
[10]  0 0⍴OPSET ⍝ invoke set up; no prompting
[11]  FMT←'BI3,X2,ZI3,3BK¯2Q< >M<(>N<)>CF9.2' ⍝ data format
[12]  T←'GALACTIC, INC.',⎕TCNL,,FMT CENTER 'SOLAR PRODUCTS'
[13]  T←T,⎕TCNL,,FMT CENTER 'SUMMER SALES FORECAST' ⍝ title
[14]  T←T,⎕TCNL,,FMT COLNAMES '/DIV/CC/PLAN /ACTUAL /VAR '
[15]  T←T,⎕TCNL,,FMT COLNAMES '/___/__/____ /______ /____ '
[16]  2 1 35 OPTOP T ⍝ skip 2 lines at top, 1 after title
[17]  T←(10⍴'-'),⎕TCNL,'NOTE: ALL DATA CONFIDENTIAL,',⎕TCNL
[18]  T←T,'      MISLEADING AND FICTITIOUS.'
[19]  3 2 OPBOT T ⍝ 3 lines before, 2 after footer
[20]  PV←1,1↓DATA[;1]≠¯1⌽DATA[;1] ⍝ 1s where div no. changes
[21]  PV OPBLOCK FMT ⎕FMT(PV×DATA[;1]); 0 1 ↓DATA) ⍝ body
[22]  OPEND ⍝ finish up
    ∇
```



Function Summary

OPALIGN	Pauses before display to screen or printer.
OPBLOCK	Displays blocks of lines that must appear on the same page. Also handles title, footer, margins, and page numbers.
OPBOT	Creates footer and margins for the page bottom.
OPDEFAULT	Restores defaults by erasing optional global variables.
OPEND	Completes the current page, erases the variable <op>, and restores printer translations.
OPOUT	Displays text at the screen, appends it to a file, or directs it to a printer. Also handles title, footer, margins, and page numbers.
OPPAGE	Completes the present page. Also handles footer, margins, and page pauses.
OPPRINT	Subroutine that sends output to a printer port.
OPPRINTFILE	Prints an APL output file on the screen, or re-directs it to a printer port or DOS file.
OPRESTART	Prompts for the page number at which to resume printing.
OPSET	Determines the output destination, creates or ties any output file, establishes any printer parameters, and creates the output parameters global variable <op>.
OPSTATE	Reviews the user-definable global variables (see "Global Variable Summary").
OPTOP	Creates title and margins for the page top.

Global Variable Summary

op	Output parameters vector (created by *OPSET*, used throughout the output facility, erased by *OPEND*).
opap	Argument to ⎕*ARBIN* when output is directed to a printer (created by *OPSET*, used by *OPOUT*, *OPPAGE*).
opb	Footer with margins (created by *OPBOT*, used by *OPOUT*, *OPPAGE*).
opd	Output destination (used by *OPSET*).
opf	Output file name (used by *OPSET*).
opl	Output lines per page (used by *OPSET*).
opm	Switch for screen display of status messages when output is not directed to the screen (used by *OPSET*).
opp	Previous page number, number of pages to print before pausing, page number justification at top, page number justification at bottom, and custom page number format phrase switch (used by *OPSET*).
oppf	Custom page number format phrase, a left argument to the system function ⎕*FNT* (used by *OPOUT*, *OPPAGE*).
oprs	Number of the page at which to resume printing (created by *OPRESTART*, used by *OPOUT*, *OPPAGE*).
opt	Title with margins (created by *OPTOP*, used by *OPOUT*, *OPPAGE*).
opw	Width of page when page numbers are right justified or centered (used by *OPSET* but can be overridden by *OPTOP*, *OPBOT*).
opx	Translate table or translation code (used by *OPSET*, *OPOUT*, *OPPAGE*).

Introduction

This chapter describes the first of three APL*PLUS/PC TOOLS supporting IRMA™, the Decision Support Interface™ from Digital Communications Associates (DCA) (formerly Technical Analysis Corporation). IRMA is a communications link between an IBM PC or IBM PC XT and an IBM 3270 network. You must install the board in your computer before using any of the IRMA-related tools.

The overall IRMA support in APL*PLUS/PC TOOLS consists of:

1. A terminal emulator that allows you to use a PC as a IBM 3278 Model 2 terminal, but with the keyboard layout of the APL*PLUS/PC System. You can use the emulator either directly from DOS or from the APL*PLUS/PC System.

2. APL printer support for the pseudo-3278 terminal using the EPSON FX series printers, even when used from DOS rather than APL.

3. A mechanism for copying files between the PC and the host under program control at about twice the speed possible using a serial connection at 1200 baud. (Refer to "The *I78XFER* Workspace".)

4. APL functions for building applications that use the IRMA Board's communications link directly. (Refer to "The *I78FNS* Workspace".)

To provide this IRMA support, APL*PLUS/PC TOOLS contains:

1. STSC's customization of the terminal emulator (Decision Support Interface) furnished with the IRMA Board, and replacement software for making further modifications with the GENX program supplied with the IRMA Board. (See "Installation of the Terminal Emulator Software".)

2. A start-up procedure that allows the STSC terminal emulator to use the downloaded APL character set (provided with the APL*PLUS/PC System) in EPSON FX printers whether used with APL or only with DOS. (See "Activating the Installed Software".)

3. A detailed explanation of the use of the APL*PLUS/PC-like keyboard for the terminal emulator. (See "Using the STSC Version of the DCA Emulator".)

4. A fully developed application workspace that can copy data or entire files between the disk drives of the PC and mainframe file systems. These files can be either DOS or APL*PLUS/PC System component files on the PC, and they can be either CMS files or APL*PLUS System component files on the mainframe. This file transfer facility also includes a "bootstrap" procedure for installing the appropriate workspace on your mainframe. (Refer to "The *I78XFER* Workspace".)

5. A workspace of APL utility functions allowing your own programs to use the communications link of the IRMA Board directly so that you can simulate a 3270-type terminal from an APL*PLUS/PC System environment. (Refer to "The *I78FNS* Workspace".)

THE *I78FNS* WORKSPACE

Introduction

The functions in the *I78FNS* workspace allow you to write APL applications that use the IRMA Board communications link to a mainframe's IBM 3274 or 3276 controller as if you were at an IBM 3278 Model 2 terminal (3278-2). Using these functions, you can:

- access the screen by row or by column
- read, modify, or write unprotected fields with automatic conversion between characters in the PC's ⎕AV with PC screen attributes and the extended buffer format of the 3278-2 screen
- use updatable character translation tables
- access all available screen information, including the extended attributes of the 3278-2 screen
- access all the status information (such as alarm, cursor position, and line synch).

These functions parallel the BASICA subroutines provided with the IRMA Board. In addition, other functions allow you to:

- access the screen by windowed areas
- translate 3278 attributes into PC attributes usable on the PC screen
- determine whether the 3278 is in APL mode, and switch to APL mode.

This chapter also includes an example of how to move mainframe data via the screen to the PC. For a more formal data transfer facility using the IRMA Board, see "The *I78XFER* Workspace". For information on the IRMA terminal emulator, refer to *I78TERM*.

THE *I78XFER* WORKSPACE

Introduction

The *I78XFER* workspace provides a means of high-speed data transfer between your PC and a mainframe able to support IBM 3270-type mainframe terminals.

The workspace is designed to provide an APL system with two levels of data transfer between the PC and the host machine. The first level allows the movement of data from a file on one machine to a file on the other.

The second level of data transfer is a protocol to allow a character vector exported from one machine to be imported on the other. The character vector is provided as an argument to the export function on the source machine and returned as the result of the import function on the target machine. The characters are transmitted as ⎕*AV* elements and the indices in ⎕*AV* are the same at each end. If your source machine and target machine use different ⎕*AV*s you need to perform a translation or else you may not end up with the same characters.

Data is transferred through the coaxial cable link furnished by the IRMA Board, using APL programs running at each end of that link and the GDDM program running on the mainframe. The corresponding APL workspace on the mainframe, *I78HOST*, can be installed using the *INSTALLI78HOST* function and the appropriate files from APL*PLUS/PC TOOLS: *I78UPFNS*, *I78MFFNS*, and *I78PLFNS*.

The mainframe workspace can be installed either under VM/CMS VS APL or the APL*PLUS/1000 System. It exists on STSC's APL*PLUS VM Service as workspace 86 *I78HOST*.

The tasks handled for you in the transmission process by the functions in this workspace include:

- the handling of the necessary blocking and deblocking of data before and after transmission
- the numbering and tracking of the blocks transmitted to ensure that all blocks have been received and reassembled in the correct order
- the use of a checksum to check that the data in a block has not been garbled in transmission
- the provision of a channel of communication independent of the data by which errors discovered by one computer are communicated to the other
- advance planning in the design to allow the tools built now for distributed applications to be used with other communications links that may become available
- the provision for cooperative processing between the mainframe and the PC.

Included in the *I78XFER* workspace are

- functions that control the actual transmission from the PC end and can activate the matching functions at the mainframe end
- functions that you can use as tools for customizing transfer functions for your particular environment, such as data packing and unpacking, translation into and out of the character vector form used for transmission, translations between ⎕*AV* tables, and so on
- functions to initiate and handle transfer of completed files of various types.

NOTE: THE I78HOST WORKSPACE CAN ALSO BE INSTALLED UNDER MVS/TSO VSAPL WHERE THE SECOND LEVEL DATA TRANSFER FUNCTIONS OPERATE CORRECTLY. THE HIGH LEVEL FILE TRANSFER FUNCTIONS ARE UNLOCKED PROTOTYPES FOR CMS FILES THAT YOU CAN ADAPT TO AN MVS FILE SYSTEM.

APL*PLUS/PC TOOLS -- WSDOC FACILITY

Introduction

The Workspace Documentation facility (WSDOC) is a simple yet powerful method for listing the contents of a workspace. You can produce the names of the functions and variables in the workspace, as well as listings of variables and unlocked functions. In addition, you can request reports that cross-reference the use of identifiers in the workspace. You can direct display to the screen or route it to an APL file.

You can obtain WSDOC output in several formats and levels of detail. You can choose the reports to display and the level of detail.

The *WSDOC* workspace contains the "operational" functions that derive the selected reports and direct them to screen or file. This chapter illustrates the available reports and documents the operational aspects of the WSDOC facility. The *WDSTATE* workspace is a separate workspace where you establish options and settings for WSDOC. Refer to "The *WDSTATE* Workspace" for information on the state setting aspects of WSDOC.

WSDOC generates the following reports:

- ENVIRONMENT Report
- System Commands Reports
  - VARS Report
  - FNS Report
  - COMMENTS Report
- FNDEF (Function Definition) Report
- CROSSREF Reports
  - VARXREF Report
  - FNXREF Report
  - LOCALXREF Report
  - GLOBALXREF Report
  - MAINFNSSUBFNS Report
  - TREES Report
  - LOCALVARS, GLOBALVARS, and COMPACT GLOBALVARS Reports

STATE

Files

```
WSFILE SAVE
OUTPUT TOSCREEN
XRFILE SAVE, ¯1 ⍝ ** Warning: no file tied to ¯1
```

Page Format

```
DEPTH 66 ⋄ WIDTH 79
MARGIN 3 3 0 0 ⍝ (top, bottom, left, right)
TITLE ''
PAGENOS 1
PAGEFNDEF MULTIPLE, ONECOL
     NOSTAMP ⋄ NODIAMOND ⋄ NOALIGNCOMMENTS
PAGEPAUSE 0
```

Content

```
NOLIST STATEOPTIONS
NONDEFAULT ENVIRONMENT
LIST VARS, FNS, FNDEF, MSGS ⋄ NOLIST LOCKEDFNS
NOFNS ''
```

Cross Reference

```
LIST LOCALXREF ⋄ PAGELXREF MULTIPLE, TWOCOL
     LIST QUADFNS IN LOCALXREF
LIST GLOBALXREF  ⋄ LABELS
     LIST QUADFNS IN GLOBALXREF
LIST MAINFNSSUBFNS
LIST TREES FOR MAINFNS, BRANCHFNS
     NOLIST QUADFNS IN TREES
     BRANCH FROM ''
     TREEHEIGHT 10
LIST GLOBALVARS FOR BRANCHFNS
LIST LOCALVARS FOR MAINFNS
NOLIST VARXREF, FNXREF, COMPACT GLOBALVARS
```

DEFAULT STATE SETTINGS

```
0 TTT3D  ⋆  Saved  16:06:57 06/25/84

02:47:04 08/20/84

     ▽ COMPUTERMOVE;A;Move;Nblank
[1]       Nblank←4-Nhits1+Nhits2
[2]       →a IF COMPUTERWINS
[3]       →a IF PERSONBLOCKED
[4]       →a IF FORK
[5]       →a IF PERSONCHECKED
[6]       →a IF TWOINAROW
[7]       FINDMOVE Moves                    ⍝ Arbitrary
[8]  a:    12 19 1 20 OUTPUT 20↑⎕A←,TRIPLE Move
[9]       A POST computerchr ⋄
          BdVec[Move]←computerchr
[10]      Nhits2←+/Hits2←Hits2∨Wins∈Move
[11]      Moves←(Move≠Moves)/Moves
[12]      Player←2 ⋄
          Nplays2←Nplays2+1
     ▽

A                 LCL    0    8←   9
BdVec                    9[←
COMPUTERWINS      FN     2
FINDMOVE          FN     7
FORK              FN     4
Hits2                    10←  10
IF                FN     2    3    4    5    6
Move              LCL    0    8    9   10   11
Moves             LCL    7   11←  11   11
Nblank            LCL    0    1←
Nhits1                   1
Nhits2                   1    10←
Nplays2                  12←  12
PERSONBLOCKED     FN     3
PERSONCHECKED     FN     5
POST              FN     9
Player                   12←
TRIPLE            FN     8
TWOINAROW         FN     6
Wins                     10
a                 LAB    2    3    4    5    6    8:
computerchr              9    9
OUTPUT            FN     8
:                        8
←                        2    3    4    5    6
```

FUNCTION LISTING WITH LOCAL CROSS-REFERENCE

A	COMPUTERMOVE	LCL←
	EXPLAIN	ARG←
	FINDMOVE	LCL←
	IF	ARG
	INTRODUCTION	LCL←
	PERSONMOVE	LCL←
	TRIPLE	ARG
ASKMOVE		FN
	PERSONMOVE	
B	EXPLAIN	LCL←
	IF	ARG
	INTRODUCTION	LCL←
BdVec	ASKMOVE	←
	COMPUTERMOVE	←
	GAMEOVER	←
	INITIALIZATION	←
	PERSONMOVE	←
	TTT3D	LCL
Board		VAR
	INITIALIZATION	
C	INTRODUCTION	LCL←
CHASTISE		FN
	ASKMOVE	
	PERSONMOVE	
CHOICES	FINDMOVE	ARG
CHR	POST	ARG
COMPUTERMOVE		FN
	TTT3D	
COMPUTERWINS		FN
	COMPUTERMOVE	
	SELFPLAY	
Cmds		VAR
	ASKMOVE	
DESCRIBE		FN
END	GAMEOVER	←
	INITIALIZATION	←
	TTT3D	LCL
EXITGAME		FN
	ASKMOVE	
EXPLAIN		FN
FINDMOVE		FN
	COMPUTERMOVE	
	PERSONCHECKED	
	SELFPLAY	
	TWOINAROW	
FORK		FN
	COMPUTERMOVE	
	SELFPLAY	
GAMEOVER		FN
	TTT3D	
HELP	ASKMOVE	LAB
HELPTEXT		FN
	ASKMOVE	
	INTRODUCTION	

XREF: A - HELPTEXT

GLOBAL CROSS-REFERENCE FOR AN ENTIRE WORKSPACE

Unlocked functions

Main functions A functions that call other functions, but are not called

TTT3D

Intermediate functions A functions that are called and that call other functions

ASKMOVE	COMPUTERWINS	FORK	INTRODUCTION	PERSONCHECKED	SELFPLAY
COMPUTERMOVE	FINDMOVE	GAMEOVER	PERSONBLOCKED	PERSONMOVE	TWOINAROW

Terminal functions A functions that are called, but do not call

CHASTISE	HELPTEXT	INITIALIZATION	STP	WAIT
EXITGAME	IF	POST	TRIPLE	

Isolated functions A functions that neither call nor are called

:SCRIBE EXPLAIN

Locked functions

Functions called as subroutines by unlocked functions

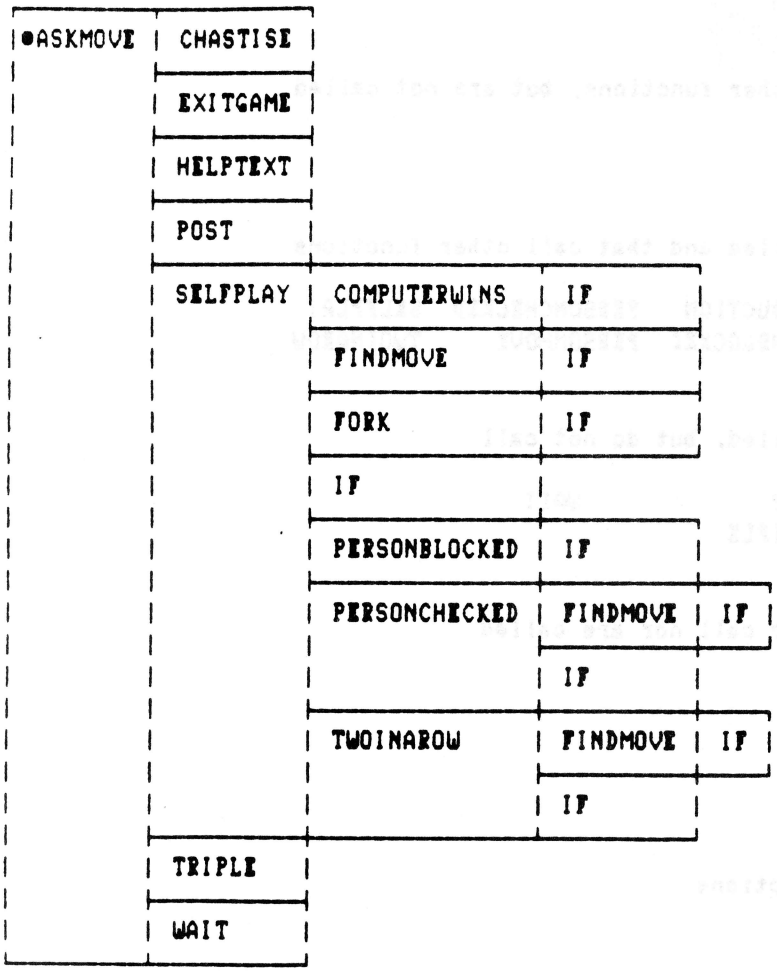
Functions not called as subroutines by unlocked functions

Global variables

Not referenced by unlocked functions in the workspace

NL

FUNCTION CLASSIFICATIONS



Objects global to the execution of <ASKMOVE>

Object	:	Referenced by				
BdVec	:	ASKMOVE				
Cmds	:	ASKMOVE				
Hits1	:	ASKMOVE	FORK	PERSONBLOCKED	PERSONCHECKED	SELFPLAY
Hits2	:	COMPUTERWINS	FORK	SELFPLAY	TWOINAROW	
Move	:	ASKMOVE	COMPUTERWINS	FINDMOVE	FORK	PERSONBLOCKED
Moves	:	ASKMOVE	SELFPLAY			
Nhits1	:	ASKMOVE	FINDMOVE	FORK	PERSONBLOCKED	PERSONCHECKED
		SELFPLAY				

FUNCTION DEPENDENCY TREE DIAGRAM

TREE: ASKMOVE

Introduction

The *WSCOMP* workspace allows you to compare the functions and variables in several workspaces. The main function is also called *WSCOMP*. It operates on file images of workspaces (called "workspace-files") created by the function *WSTOFILE* (see the *WSTOFILE* workspace). The principal basis of comparison is the function definitions and the values of the variables, but you can also obtain lists of functions or variables that are identical in both workspaces or that are in one workspace only.

For each function that is different in the two workspaces, *WSCOMP* displays only those function lines that are different or missing from the function in the other workspace. *WSCOMP* does not display lines that are identical, even if they are shifted in line number because of insertions or deletions.

When a function is locked in one workspace, *WSCOMP* notes that fact and suppresses any display of an unlocked counterpart in the other workspace.

Certain types of comments (those beginning with ⍝⍝) can be sheltered from comparison and may be appropriate places to store configuration information that may naturally vary from workspace to workspace.

Whenever *WSCOMP* displays function or variable differences, you may elect to have timestamps printed indicating when those functions and variables were stored in the workspace-files.

You can direct *WSCOMP* output to the screen or to an APL file. All output is handled through functions from the *OUTPUT* workspace, so the capability of that facility to redirect output is available.

WSCOMP can loop through many paired comparisons, so that one "primary" workspace is compared against several "secondary" workspaces, and then another "primary" against its list of "secondary" workspaces, and so on.

State settings in the workspace determine *WSCOMP* options. You can review and change these settings with the full-screen *WCSTATE* function. The possible choices follow.

- You can direct your output to screen or APL file.
- You can compare functions, variables, or both.
- You can elect to display lists of the chosen objects (functions or variables) that are equal in both workspaces if you chose to compare that type of object.
- For whatever objects you choose to compare, you can elect to display a list of the chosen objects in the primary workspace not present in the secondary, or vice versa. The setting *LIST ABSENT FNS IN PRIMARY* means to list all functions in the primary workspace that are absent from the secondary.
- You can choose to check if the referenced files are actually workspace-files.
- You can choose to ignore specially tagged comments (⍝⍝) in function comparisons.
- You can choose to display file store timestamps for functions and variables that differ.

In addition to comparison of entire workspaces, the stand-alone function *FNCOMP* compares two individual functions. Either of the functions can be in the active workspace or in an APL component file.

Workspace: *WSCOMP* **Function**: *FNCOMP*

Relates to: *WFFIND*, *WSCOMP*

A stand-alone comparison of two functions that are workspace or file resident (or one of each).

Syntax

fnspec1 *FNCOMP* fnspec2

The arguments fnspec1 and fnspec2 are either the names of functions in the active workspace; or, if numeric, a file tie number, component number, and optional passnumber. Such a file component must contain a visual representation (⎕*VR*) of a function. In general, no specific file structure is assumed, but if a file is a workspace-file (created by the *WSTOFILE* function), then you can use the function *WFFIND* to determine the component number for a given function.

Effect

FNCOMP displays on the screen the function lines that are different for the functions being compared. The function specified by the left argument appears first, and a dashed line separates the two functions.

Examples

Compare two functions in the workspace:

```
'MONTHLYREPORT' FNCOMP 'YEARLYREPORT'
```

Compare a function in the workspace with the function in component 35 of the file tied to 90:

```
'REPORTS' FNCOMP 90 35
```

Compare a function in the workspace to its representation in a workspace-file tied to 90:

```
'REPORTS' FNCOMP 90 WFFIND 'REPORTS'
```

Compare a function found in workspace-files tied to 90 and 92:

```
(90 WFFIND 'REPORTS') FNCOMP 92 WFFIND 'REPORTS'
```

Sample WSCOMP Output

```
Workspace comparison of <0 OUTPUT> to <2 OUTPUT>
Printed 10/10/84 at 13:13:30


3 functions in <0 OUTPUT> not in <2 OUTPUT>:

  CENTER        COLNAMES      RWTD

0 functions in <2 OUTPUT> not in <0 OUTPUT>:

      ⋮
===== Function -- OPRAW

      in <0 OUTPUT>

   ∇ NL OPRAW X;⎕ALX;⎕IO
[3]  ⍝ <X>  is a character array which is appended to the output file, printed
[4]  ⍝   or displayed on the screen, as appropriate
[12]  ⎕ALX←'⎕ERROR''ATTN''' ◊ ⎕IO←1
[14]  ⎕ERROR(op[3]≤+/op[4 5])/'OUTPUT ERROR: space needed for title and footer
      exceeds page length'
[15]  ⎕ERROR(82≠⎕DR X)/'CHARACTER RIGHT ARGUMENT ONLY'
[16]  →op[9]↓L1 ◊ →(⍴⍴opx)↓L1 ◊ X←opx[⎕AV⍳X] ⍝ Translate before printing
[17] L1:op[7]←op[7]+NL ◊ op[6]←op[3]|op[6]-NL ◊ →(op[2]∈ 1 4)⍴L3 ⍝ if to screen
       or printer

      in <2 OUTPUT>

   ∇ NL OPRAW X;TN;⎕ALX;⎕IO
[3]  ⍝ <X>  is a character array which is appended to the output file
[4]  ⍝   or printed on the screen, as appropriate
[13]  ⎕IO←1 ◊ ⎕ERROR(op[3]≤+/op[4 5])/'OUTPUT ERROR: space needed for title and
       footer exceeds page length'
[14]  ⎕ALX←'⎕ERROR''ATTN''' ◊ TN←' ',1↑0⍴X ◊ →op[9]↓L1 ◊ →(⍴⍴opx)↓L1 ◊ X←opx[⎕A
      V⍳X] ⍝ Translate before printing
[15] L1:op[6]←op[3]|op[6]-NL ◊ →(op[2]∈ 1 4)⍴L3 ⍝ if to screen or printer
      ⋮
12 functions identical in both workspaces:

  EXPLAIN       OPALIGN       OPBOT         OPSETAP
  INSTALL       OPAPPEND      OPEND         OPTOP
  INSTALLSNUFF  OPBLOCK       OPPAGE        SUMMARY

0 variables in <0 OUTPUT> not in <2 OUTPUT>:


0 variables in <2 OUTPUT> not in <0 OUTPUT>:


4 variables identical in both workspaces:

  DATA                  INSTALLOPEN
  INSTALLINSTRUCTIONS   wsid
```